Mechanical Systems and Signal Processing 165 (2022) 108284

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

journal homepage: www.elsevier.com/locate/ymssp

# Real-time model calibration with deep reinforcement learning

Yuan Tian [a,1], Manuel Arias Chao [a,1], Chetan Kulkarni [b,c], Kai Goebel [d], Olga Fink [a,*]

[a] ETH Zürich, Switzerland
[b] KBR Inc., United States of America
[c] NASA Ames Research Center, United States of America
[d] Luleå University of Technology, Sweden

ARTICLE INFO

Communicated by Y. Lei

*Keywords:*
Model calibration
Reinforcement learning
Model-based diagnostics
Deep learning

ABSTRACT

The real-time, and accurate inference of model parameters is of great importance in many scientific and engineering disciplines that use computational models (such as a digital twin) for the analysis and prediction of complex physical processes. However, fast and accurate inference for processes of complex systems cannot easily be achieved in real-time with state-of-the-art methods under noisy real-world conditions with the requirement of a real-time response. The primary reason is that the inference of model parameters with traditional techniques based on optimization or sampling often suffers from computational and statistical challenges, resulting in a trade-off between accuracy and deployment time. In this paper, we propose a novel framework for inference of model parameters based on reinforcement learning. The proposed methodology is demonstrated and evaluated on two different physics-based models of turbofan engines. The experimental results demonstrate that the proposed methodology outperforms all other tested methods in terms of speed and robustness, with high inference accuracy.

## 1. Introduction

Inference of computational model parameters from real-time measurements can be referred to as *model calibration* [1]. Model calibration aims to both obtain model parameters that are theoretically plausible and generate model predictions that fit the observations. The inferred model parameters often represent physical quantities that are not directly observable or observed, i.e., they are not directly obtained from sensor measurements. Therefore, the inference of physics-based model parameters enables one to understand the underlying reasons for a discrepancy between physics-based model predictions and observations, i.e., the *reality gap* (see Fig. 1). This is of particular relevance for scientific and engineering disciplines where one is interested in improving the physics-based models analytically or explaining the observed processes in light of a given physics-based model structure. Applications can be found in multiple areas, including geology [2], climatology [3], biology [4], health [5], finance [6,7], cognitive science [8], mechanical engineering [9], and applied physics [10].

A particularly important field of application aiming at a reasoned analysis of discrepancies between model predictions and observations is **model-based system health diagnostics** of safety-critical engineered systems. Diagnostics involves detecting when a fault occurs, isolating the root cause, and identifying the extent of the damage [11]. In model-based health diagnostics, the discrepancy between model and observation is interpreted as a deteriorated or anomalous response of the system. Model-based health diagnostics addresses the diagnostics problem by inferring the value of model parameters, representing the health condition

* Corresponding author.
*E-mail address:* ofink@ethz.ch (O. Fink).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.ymssp.2021.108284
Received 28 October 2020; Received in revised form 14 July 2021; Accepted 24 July 2021
Available online 16 August 2021

**Nomenclature**

**Nomenclature for Model Calibration**

| | |
|---|---|
| $\hat{x}_t$ | Model output at time step t |
| $\omega_t$ | Flight condition at time step t |
| $\theta_t$ | Model dynamical parameters for time step t |
| $x_t$ | Sensor measurement at time step t |

**Nomenclature for Reinforcement Learning**

| | |
|---|---|
| $\alpha_3$ | Energy decreasing speed parameter |
| $\beta$ | Temperature parameter |
| $\gamma$ | Discounted factor |
| $\mathcal{D}$ | Replay Buffer |
| $\mathcal{H}$ | Entropy |
| $\rho_\pi$ | Policy |
| $\tau$ | Trajectory |
| $a$ | Action |
| $c_\pi$ | Cost Function |
| $J(\pi)$ | Discounted cumulative reward |
| $L$ | Lyapunov function |
| $P$ | Transition Function |
| $s$ | State |
| $s'$ | Next State |
| $t$ | Time step |

**Nomenclature for Neural Network**

| | |
|---|---|
| $\beta$ | Positive Lagrange multiplier controls the importance of stability guarantee |
| $\lambda$ | Positive Lagrange multiplier controls the importance of policy entropy |
| $\overline{\phi_L}$ | Target Lyapunov network parameters |
| $\phi_L$ | Lyapunov network parameters |
| $\pi_\theta^*$ | The mean of the current policy output distribution |
| $\pi_\theta$ | Policy Network |
| $\tau$ | Target network soft-update parameter |
| $\theta$ | Policy network parameters |
| $J(\beta)$ | Objective function for $\beta$ |
| $J(\lambda)$ | Objective function for $\lambda$ |
| $J(\pi)$ | Objective function for Policy network |
| $J(L_c)$ | Objective function for Lyapunov network |
| $L_c^{target}$ | Target Lyapunov network |
| $L_c$ | Lyapunov network |
| $s_{near}$ | A near state to state $s$ |

**Others**

| | |
|---|---|
| $\alpha$ | Model bias intensity |
| $\alpha_\eta$ | Gaussian noise intensity |
| $F$ | Dynamic system, dynamic model or surrogate model |
| $N$ | Gaussian noise |
| $SNR_{db}$ | Signal to noise ratio |

of the sub-components of a system that make the physics-based model predictions fit the observations. In this way, anomalies in the system's behaviour are detected and characterized by the value of model parameters.

Because of the relevance of model calibration in applications such as model-based diagnostics, it is important that model calibration provides accurate inference of the model parameters while being robust to uncertainty in the observations and the physics-based model structure. Calibration in real-world scenarios faces **computational and statistical difficulties**. Computational

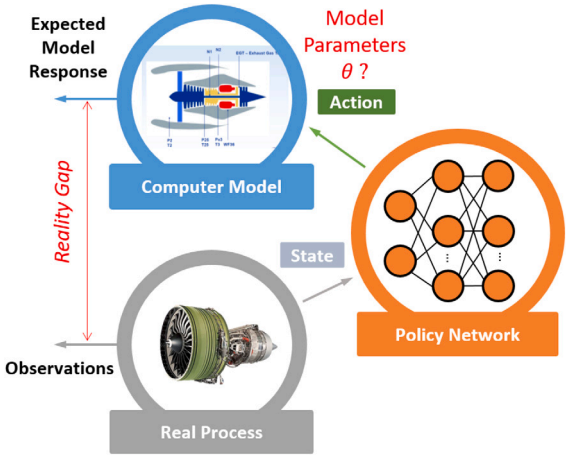


**Fig. 1. Calibration of physics-based models** aims to infer model parameters that make the physics-based model response follow the observations, thus reducing the *reality gap*. In this work, a reinforcement learning algorithm is used to obtain a neural network policy that bridges the gap between physics-based model predictions and observations in real time.

issues are related to the need for running time-consuming simulations using optimization and inference techniques that generally imply a trade-off between inference accuracy and computation time. Scaling the method to **complex dynamic models** (such as for example flow field calculations), **high dimensional spaces** and **large datasets** further exacerbate the problem. Statistical issues arise from (a) the incompleteness of the model representation, (b) the existence of multiple solutions, i.e., *confounding solutions* that match the observations, and (c) the uncertainty of the observations. Some safety-critical applications, such as model-based diagnostics of aircraft engines considered in this paper, require an inference of the model parameters that is at the same time accurate, robust and is available in real time to enable a fast and reliable state assessment. The necessity of fulfilling all of these requirements at the same time makes the development of methods for reliable dynamical model calibration challenging.

Several methods have been proposed to address the problem of dynamical model calibration. When the physics-based model structure is well founded on known physical principles (e.g., aircraft thermodynamic engine models), the majority of the available methods for parameter inference are estimation approaches developed in the fields of **optimal control** [12] and **statistics** [13]. Some examples of popular estimation methods include iterative reweighted least squares schemes [14], unscented Kalman filters (UKF) [15–17], particle filters [18] or Bayesian inference methods using Markov chain Monte Carlo [5,14] and Gaussian Process [1,14]. Approaches of this type scale relatively well to high-dimensional calibration problems and, with their probabilistic nature, handle observation noise reasonably well. These estimation methods have achieved good results in practical applications and are considered as state-of-the-art methods in several applications, including model-based diagnostics. Yet, despite these attractive properties, they all suffer, at least to some degree, from various model computational and data statistical difficulties in real-world scenarios. In particular, this is because estimation with these methods involves multiple evaluations of the computational model, which makes them unsuitable for real-time calibration of complex models or large datasets. Moreover, these methods are particularly affected by the inadequacy of the physics-based model structure, resulting in an inaccurate characterization of the reality gap.

More recently, **data-driven approaches** have been proposed to calibrate physics-based models. Aiming to avoid time-consuming simulations of previous calibration methods and achieve real-time model calibration, some researchers have proposed alternative approaches to probabilistic formulation of the calibration problem. The most common approach is to address the calibration problem as a supervised learning problem [6]. In this case, a neural network algorithm is trained in the inverse relation between the observations and the model parameters. Although these methods provide a real-time calibration approach (only a forward pass over a neural network is required at deployment time), the accuracy of the methods is strongly dependent on the representative quality of the training datasets. As a result, this model calibration approach is not able to adapt to new scenarios without re-training. To mitigate this limitation, an exhaustive mapping of possible system responses under different flight conditions and values of model parameters is required. In practice, in high-dimensional calibration problems with a large range of flight conditions, an exhaustive mapping is infeasible. In addition, such methods can exhibit poor performance in scenarios involving noisy observations, which can limit their implementation in practical applications when no noise mitigation measures are taken into consideration in the learning process.

Because of the issues mentioned above, the real-time, robust, and accurate inference of physics-based model parameters of complex engineered systems remains challenging. However, recent developments in **model-free reinforcement learning (RL)** have fostered a great deal of progress in addressing similar challenges in control problems [19]. In fact, RL has proven to be effective in finding optimal control policies for non-linear stochastic systems when the dynamics are either unknown or affected by severe uncertainty [20], including complicated robotic locomotion and manipulation [21–23]. The policies learnt via RL have the ability to adapt to new scenarios and scale well to large-scale problems at run time. In fact, the decision-making of reinforcement learning can take place through a learned policy without any further optimization or model evaluation, which overcomes the inference speed

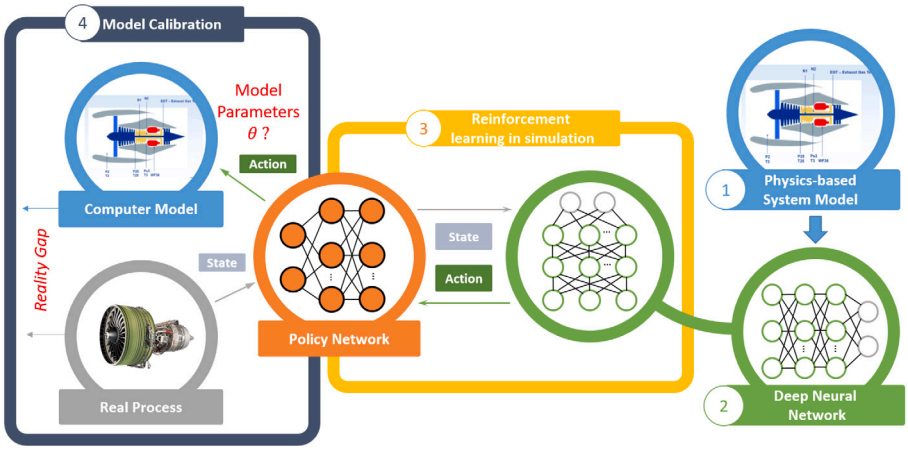


**Fig. 2. Creating a calibration policy:** Step 1, we identify the parameters of the physics-based model we intend to calibrate. Step 2 (optional), we create a deep neural network (DNN) that models the complex system dynamics. Step 3, we train a control policy using the physics-based model or the DNN model. **Implementation stage:** Step 4, we deploy the trained policy for real-time model calibration.

problem at deployment time. The learned policy can take the form of a neural network or use other function approximation methods. Therefore, model-free RL [24] is a compelling alternative to traditional inference methods for physics-based model calibration.

In this work, we propose a novel formulation of the calibration problem as a **tracking problem** that is modelled by a Markov decision process. Based on this formulation, we apply Lyapunov-based maximum entropy deep reinforcement learning to train an agent that controls the physics-based model parameters to keep the model response matching the observations. In order to overcome the traditional high variance of RL and achieve better robustness to observation uncertainty and model inadequacy, we propose a novel constrained Lyapunov-based actor–critic (CLAC) algorithm. The proposed CLAC algorithm adds constraints on the stability of the policy network and is an extension of the Lyapunov-based actor–critic (LAC) algorithm [25].

Without any knowledge of the physics-based model or simulator, the agent explores a large range of possible dynamical responses of the system resulting in good and bad rewards. As a result, the agent is able to exploit the dynamics of the model and produce a robust control (i.e., calibration) logic. Therefore, the proposed framework overcomes the difficulties of traditional optimal control methods, data-driven approaches and current RL algorithms. It provides: (a) an accurate real-time dynamical calibration, (b) a policy that can adapt to new scenarios without having been specifically trained on them, (c) scalability to more complex systems and high-dimensional spaces, and (d) robustness to observation and model uncertainty.

The proposed framework is summarized in Fig. 2. In the first step, we identify the parameters of the physics-based model that are subjected to inference. In the second step, we use a physics-based model or, alternatively, a surrogate model, in our case a deep neural network (DNN) that emulates the expected system response for measured properties (i.e., observations). In the third step, we use the DNN model to train the calibration policy network via RL. At deployment time, the trained calibration policy is directly deployed to obtain the physics-based model parameters at run time (step 4). The resulting calibration policy is computationally efficient at run time. Most importantly, the calibration policy is robust to uncertainties both in the observations and the physics-based model. The proposed methodology is demonstrated and evaluated on two different physics-based models of a turbofan engine: the Advanced Geared Turbofan 30,000 (AGTF30) and Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) from NASA.

The contribution of this paper is two-fold: (1) We propose a solution to the problem of **real-time dynamic calibration of physics-based models**. In particular, we present a general reinforcement-based model calibration framework that enables real-time inference of system model parameters with a single forward pass through a neural network. While the performance of the framework is demonstrated on two turbofan engine models in this research, it could be easily applied to any system model. Furthermore, the proposed framework does not require any labelled data or demonstration for training. This is in line with the requirements of industrial scenarios where such labels are typically lacking. (2) From the methodological perspective, we propose the **constrained Lyapunov-based actor–critic (CLAC) algorithm**, which provides more action stability, especially on parameter tracking problems, compared to the state-of-the-art LAC reinforcement learning algorithm. This makes the proposed approach robust to noise and high variability, which is again in line with requirements of industrial applications.

## 2. Preliminaries

In this section, we briefly review the basic concepts and notations related to reinforcement learning which is in the focus of the proposed framework.

### 2.1. Reinforcement learning

Reinforcement learning is a sub-field of machine learning that focuses on how an agent interacts with the environment to achieve a specific goal. The environments are typically stated in the form of a Markov decision process (MDP), which provides a mathematical description of decision-making processes. Under the right problem formulation, MDPs can be useful for solving optimization and inference problems, such as the one described above for physics-based model calibration, via reinforcement learning. The details of the MDP formulation of physics-based model calibration will be discussed in Section 3.

In conventional reinforcement learning, an agent is trained to interact with the environment and seek rewards on the basis of its actions. The agent receives a successor state $s_{t+1}$ from the environment as feedback in response to a decision (i.e., action $a_t$) taken at time-step $t$. The goal is to find a policy $\rho_\pi$ that maximizes the discounted cumulative reward $J(\pi)$ [26], which is given by the following expression:

$$J(\pi) = \mathbb{E}_{\tau\sim\rho_\pi} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor.

### 2.2. Maximum entropy RL

The maximum entropy reinforcement learning framework considers a more general objective, aiming to learn a stochastic policy which jointly maximizes the expected discounted cumulative reward and its expected entropy $\mathcal{H}(\pi(\cdot|s_t))$ [27]:

$$J(\pi) = \mathbb{E}_{\tau\sim\rho_\pi} \sum_{t=0}^{\infty} \gamma^t [r(s_t, a_t) + \beta\mathcal{H}(\pi(\cdot|s_t))], \tag{2}$$

where $\beta$ is the temperature parameter that controls the stochasticity of the optimal policy over the reward. Therefore, the resulting stochastic policies balance the exploration–exploitation trade-off and add robustness to the policy.

### 2.3. Stability guaranteed RL

The maximum entropy reinforcement learning framework can also include a closed-loop stability guarantee of the system dynamics. Such a stability guarantee is particularly relevant when dealing with control problems in real-world applications. Recently, the Lyapunov-based actor–critic (LAC) method [25], implementing a stability guarantee, showed state-of-the-art performance on tracking tasks. From a control-theoretic perspective, the task of tracking can be addressed ensuring that the closed-loop system is asymptotically stable. In other words, starting from an initial point, the trajectories of states always converge to a single point or reference trajectory. Therefore, in [25], a stability-guaranteed reinforcement learning framework is proposed under the following definition of stability:

*Stability definition.* Suppose $c_\pi(\cdot)$ is the cost function, $c_\pi : S \to \mathbb{R}_+$. The system is said to be mean square stable (MSS) if $\lim_{t\to\infty} \mathbb{E}_{s_t} c_\pi(s_t) = 0$ holds for any initial condition $s_0$.

Under this definition, the stability objective is given by Eq. (3). The stability objective defines an energy decreasing condition that drives the trajectory asymptotically to the null space of the cost function, producing predictable behaviour of the agent. Here, we use the Lyapunov function to denote the system's energy, so that the state goes in the direction of decreasing the value of the Lyapunov function and eventually converges to the origin or a sub-level set of the Lyapunov function.

$$\mathbb{E}_{s\sim\tau}(\mathbb{E}_{s'\sim\rho_\pi} L(s') - L(s)) \le -\alpha_3 \mathbb{E}_{s\sim\tau} c_\pi(s) \tag{3}$$

where the $\alpha_3$ term controls the energy decreasing speed.

## 3. Proposed framework - Calibration policy

### 3.1. Model calibration defined as tracking problem

In this work, we formulate the real-time model calibration problem as a tracking problem, which is modelled by an MDP, and use reinforcement learning to find the optimal tracking policy. We aim to control/infer the model dynamical parameters ($\theta_{t+1}$) to let the model output ($\hat{x_{t+1}}$) match the real sensor measurement ($x_{t+1}$). However, to solve model calibration by reinforcement learning is not trivial since it is not a conventional control task. To establish a congruence between the model calibration task and the more classical driver tracking task: a driver (equivalent to a computational model) needs to infer the operation (equivalent to physical parameters) of another driver (equivalent to real sensor measurement) in front. To solve this problem, the driver only needs to keep tracking the path of the car in front. Once driver is able to track the front car, the operation he performs, will be the operation he would like to infer. The rationale behind this solution strategy is that learning to track observations of a real system response ($x_t$) by changing the model parameters ($\theta_t$) results in a control policy that makes the physics-based model yield a sound approximation of the physical process ($\hat{x}_t$), i.e., reducing the reality gap. Consequently, the tracking policy also serves as a calibration policy [28].

Under a tracking solution strategy, the MDP describing the problem is given as the tuple $(s, a, r, P, \rho)$, where state ($s$) comprises the current model output $\hat{x}_t$, the target value of the system response (observations of the real system) $x_{t+1}$, and the flight condition

$w_{t+1}$, i.e., $s_t = [\hat{x}_t, x_{t+1}, w_{t+1}]$. The action ($a$) defines the model parameters that need to be calibrated, i.e., $a_t = \theta_t$. The reward/cost function $r(s, a)$ evaluates how good the tracking is. The state transition probability function ($P(s'|s, a)$) corresponds to the dynamics of the system that can be modelled by a physics-based or a surrogate model.

In order to speed up the learning process of the RL algorithm, a discrete time counterpart of the physics-based model $F$ is used. The resulting dynamical system $F$ or a surrogate simulator is modelled by a deep neural network that approximates the dynamic transition equation describing how the expected system response changes given the current observations $\hat{x}_t$, the flight conditions $w_{t+1}$, and model parameters $\theta_{t+1}$, resulting in:

$$\hat{x}_{t+1} = F(w_{t+1}, \hat{x}_t, \theta_{t+1}) \tag{4}$$

For the tracking problem there is, therefore, a desired state that we would like the system to be in at each time step, i.e., $x_{t+1}$. The task of the agent is to find a control policy $\theta_{t+1} = \pi(\hat{x}_t, x_{t+1}, w_{t+1})$ that minimizes the cost based on a distance metric representing the reality gap of the physics-based model. Here we use the mean squared error (MSE) between the model output $\hat{x_{t+1}}$ and the real system measurement $x_{t+1}$ as the reward signal. In particular, given the dynamical system above and a target system trajectory (i.e, observations), we train the control policy to keep the simulator output matching the real system output by maximizing the cumulative reward as defined in Eq. (1).

### 3.2. State space and action space

The state $s_t$ describes the current sensor information of the system, the target sensor information of the system and the flight conditions, which is defined as $s_t = [\hat{x}_t, x_{t+1}, w_{t+1}]$. And the action describes as the system parameters need to be calibrated, i.e. degradation parameters, which is defined as $a_t = [\theta_{t+1}]$.

### 3.3. Learning algorithm

In this work, we adopt Lyapunov-based actor–critic (LAC) [25] as the learning algorithm, which is based on the soft actor–critic (SAC) [29] algorithm and incorporates a stability guarantee objective. The stability guarantee objective enables a control policy that stabilizes the system in the case of interference by unseen disturbances or uncertainties in the system dynamics. Most importantly, the LAC algorithm has been show to yield the best performance on tracking problems [25].

Based on the maximum entropy actor–critic framework, LAC uses the Lyapunov function $L_c$ as the critic in the policy gradient formulation. The objective function of $J(L_c)$ is given as follows:

$$J(L_c) = \mathbb{E}_{(s,a)\sim D}\left[\frac{1}{2}(L_c(s, a) - L_c^{target}(s, a))^2\right] \tag{5}$$

$$L_c^{\text{target}}(s, a) = c + \gamma \arg\max_a L_c(s', a) \tag{6}$$

where $L_c^{target}$ is the approximation target for $L_c$ as typically used in RL methods [30,31]. $L_c^{target}$ has the same structure as $L_c$, but the parameter is updated through exponentially moving average of weights of $L_c$ controlled by a hyperparameter $\tau$.

The objective function for the policy network is given by:

$$J(\pi) = \mathbb{E}_D[\beta[\log(\pi_\theta(f_\theta(\epsilon, s)|s))] + \lambda(L_c(s', f_\theta(\epsilon, s')) - L_c(s, a) + \alpha_3 c)] \tag{7}$$

where $\pi_\theta$ is parameterized by a neural network $f_\theta$, and $\epsilon$ is an input vector consisted of Gaussian noise. The $\mathcal{D} \doteq \{(s, a, s', c)\}$ is the replay buffer for storage of the MDP tuples. In the above objective, $\beta$ and $\gamma$ are positive Lagrange multipliers which control the relative importance of policy entropy versus the stability guarantee. As in [32], the entropy of policy is expected to remain above the target entropy $\mathcal{H}_t$ [32], here we adopt the same strategy. The value of $\beta$ is adjusted through the gradient method, thereby maximizing the objective:

$$J(\beta) = \beta\mathbb{E}_{(s,a)\sim D}[\log(\pi_\theta(a|s)) + \mathcal{H}_t] \tag{8}$$

$\lambda$ is adjusted by the gradient method, thus maximizing the objective:

$$J(\lambda) = \lambda(L_c(s', f_\theta(\epsilon, s')) - L_c(s, a) + \alpha_3 c) \tag{9}$$

Under conditions of high sensor noise and simulator bias resulting from an incomplete representation of the system model (i.e., irreducible reality gap), the policy network can exhibit large variance. Such a situation is undesirable in many real-world applications where it is important to obtain a stable or smooth action over time. Therefore, in order to stabilize the action, we introduce the constrained Lyapunov-based actor–critic (CLAC) algorithm, a modification of the LAC, which significantly improves the action stability under model uncertainty and sensor noise. In CLAC, the objective function has an additional term that aims to obtain a policy network that has similar optimal action when given a similar or near state ($s_{near}$) and is given by:

$$\begin{aligned} J(\pi) = &\mathbb{E}_D[\beta[\log(\pi_\theta(f_\theta(\epsilon, s)|s))]+ \\ &\lambda(L_c(s', f_\theta(\epsilon, s')) - L_c(s, a) + \alpha_3 c)+ \\ &\alpha\|\pi_\theta^*(s) - \pi_\theta^*(s_{near})\|] \end{aligned} \tag{10}$$

where $\alpha$ is a positive Lagrange multiplier, and $\pi_\theta^*(s)$ outputs the action with the largest probability. In our case, we use the adjacent time space state $s_{t+1}$ or $s_{t-1}$ to approximate $s_{near}$.

The entire procedure for training the proposed constrained Lyapunov actor–critic is outlined in Algorithm 1 and the hyper-parameter settings can be found in Table 1

It is worth mentioning that in our experiments the policy is first trained in the simulation environment and then frozen for evaluation.

**Algorithm 1:** Constrained Lyapunov-based actor–critic (CLAC)

Input hyperparameters, learning rates $\alpha_{\phi_L}$, $\alpha_\theta$
Randomly initialize a Lyapunov network $L(s,a)$ and policy network $\pi(a|s)$ with parameters $\phi_L$, $\theta$ and the Lagrange multipliers $\beta$, $\lambda$
Initialize the parameters of target network with $\overline{\phi}_L \leftarrow \phi_L$
**for** each iteration **do**
  Sample $s_0$ according to $\rho$
  **for** each time step **do**
    Sample $a_t$ from $\pi(s)$ and step forward
    Observe $s_{t+1}$, $r_t$ and store $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
  **end for**
  **for** each update step **do**
    Sample minibatches of transitions from $\mathcal{D}$ and update $L$, $\pi$ and Lagrange multipliers with gradients
    Update the target networks with soft replacement:

$$\overline{\phi}_L \leftarrow \tau\phi_L + (1-\tau)\overline{\phi}_L \tag{11}$$

  **end for**
**end for**

## 4. Experiments

The proposed framework is demonstrated and evaluated on two different physics-based models focusing on the diagnostics of turbofan engines. The two case studies explore different aspects of real-world calibration problems. Case study #1 corresponds to a one-dimensional calibration problem ($d = 1$) under a wide range of real (i.e., noisy) flight conditions from a small fleet of ten units ($N = 10$). Case study #2 is a more complex system that explores complex failure modes affecting four components of the system simultaneously ($d = 4$). Therefore, case study #2 explores a calibration problem under complex system responses. In contrast to case study #1, case study #2 contains only flight condition from one single unit and, consequently, has a more limited range of flight conditions. More details about the simulator and flight condition data can be found in Appendix.

The performance of the proposed CLAC method is evaluated and compared to two alternative calibration models: a unscented Kalman filter (UKF) and a supervised end-to-end mapping with deep learning algorithm (E2E). The evaluation also covers variants of case study #1 designed to evaluate the robustness of the different methods to uncertainty in the observations and system model predictions.

### 4.1. Neural network architectures and hyper-parameters

The proposed framework requires three neural networks: policy network, Lyapunov network and surrogate network of the dynamic model.

For the policy network, we use a fully-connected multi-layer perceptron (MLP) with two hidden layers of 256 units, outputting the mean and standard deviations of a Gaussian distribution. We adopt the invertible squashing function technique as proposed in [32] to the output layer of the policy network. For the Lyapunov network, we use a fully-connected MLP with two hidden layers of 256 units, outputting the Lyapunov value. All the hidden layers use leaky-ReLU [33] activation function.

The system dynamics (surrogate network of the dynamic model) is approximated with an MLP with four layers ($L = 4$). The hidden layers have 100 units ($n^1 = n^2 = n^3 = 100$). The output layer has the dimension of the sensor reading vector (i.e. $n^L = n$). *ReLU* activation function was used throughout the hidden layers. For the output layer $\sigma^L = I$ is the identity.

The optimization of the networks' weights was carried out with mini-batch stochastic gradient descent (SGD) and with the *Adam* algorithm [34]. *Xavier* initializer [35] was used for the weight initializations. Most of the parameters setting are according to the original LAC setup [36], and the target entropy is based on the SAC result [29,32]. Table 1 provides a detailed overview of the hyperparameters used for the experiments.

**Table 1**
LAC and CLAC hyperparameters.

| Hyperparameters | Batch size | LR-Actor/Critic | Target entropy | $\tau$ | $\gamma$ | $\alpha_3$ | Initial $\beta$ | $\lambda$ |
|---|---|---|---|---|---|---|---|---|
| Value | 256 | 1e−4/3e−4 | −d | 0.005 | 0.99 | 1 | 2 | 0.1 |

**Table 2**
Overview of the inference performance given by the RMSE between the inferred model parameters and the ground truth with UKF, E2E, and CLAC approaches on complete test trajectories. Best performance is shown in **bold**. [1] The analysis with the E2E model was not performed in Case Study #2 in light of the poor results in Case Study #1.

| Method | Case Study #1 | Case Study #2 |
|---|---|---|
| UKF | 3.42e−04 | 3.51e−03 |
| E2E | 1.36e−03 | – |
| CLAC | **3.30 ± 0.38e−04** | **2.50e−03** |

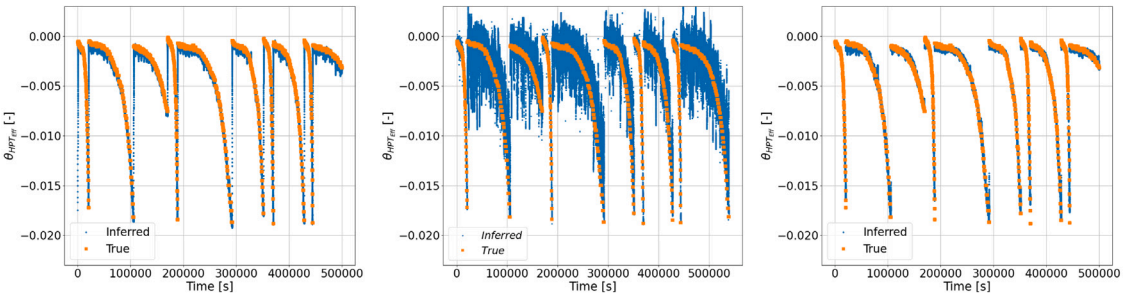


**Fig. 3.** Inferred (blue dots) and ground truth (orange squares) traces of $\theta$ in Case Study #1 with UKF (left), E2E (middle), and CLAC (right) approaches. The $\theta$ values for the ten units are stacked one after the other, generating a single time sequence. Each discontinuity corresponds to the beginning of a new unit. The UKF solution proves a good match to the ground truth with low bias and variance. It is observed that only at the beginning of each unit, the UKF predictions show a large bias. Estimations with the E2E model show a large variance, in particular for units with long degradation profiles. The CLAC method demonstrates a very good match to the ground truth.

## 5. Results

The aim of the proposed framework is to enable accurate, real-time, and robust model calibration for complex systems and large-scale problems. Therefore, in this section, the performance of the proposed method is analysed based on six evaluation criteria: (1) inference accuracy, (2) computational cost, (3) robustness to system model uncertainty, (4) robustness to observation noise, (5) scalability to more complex systems, and (6) tracking accuracy.

**Inference accuracy.** The primary objective of model calibration is to infer the values of the model parameters $\theta$. From the application perspective of model-based diagnostics, this objective corresponds to inferring the true underlying degradation parameters. Therefore, we compare the estimated degradation parameters ($\hat{\theta}$) with the ground truth and report the inference accuracy in the form of the root mean square error (RMSE). Table 2 shows the inference performance of the unscented Kalman filter (UKF), end-to-end mapping (E2E), and the proposed method (CLAC) in both datasets. With the lowest RMSE, the policy obtained with CLAC shows the best overall performance in both case studies. The improvement is particularly significant under complex fault modes (i.e., Case Study #2). The E2E model yields the worst overall performance in Case Study #1, which highlights the limitations of supervised learning in cases where the flight condition dataset for training is not fully representative of the test conditions. Fig. 3 shows the inferred unobserved model parameters $\hat{\theta}$ obtained with the three methods in Case Study #1. It is worth mentioning that unlike the end-to-end mapping, which needs the ground truth degradation parameters for training, our framework does not need any prior knowledge about the degradation parameters. This makes the approach more flexible and more applicable to real scenarios.

Since reinforcement learning policies might suffer from a high variance [37], we evaluated the reproducibility of the proposed method by training our agent over five different seeds. As shown in Fig. 4, we observe that our agent shows a good reproducibility on both tasks.

**Computational cost.** One crucial aspect of the proposed method is the ability to perform calibration in real time which is a crucial requirement for real applications. Therefore, we evaluate the time required to perform inference of the model parameters at deployment. Table 3 reports the average times required to calibrate a single sample and the total training time with the three methods. In terms of deployment computational cost, the proposed method provides a speed up of $\times 150$ compared to the UKF. Concretely, inference with the proposed CLAC method takes around 40 ms using a CPU thread. This deployment speed is comparable to the E2E model as both methods only require a forward pass over a deep neural network. In contrast, the UKF needs to perform $2\times(2\times d+1)$ model evaluations, which for Case Study #1 amounts to 6 s. The CLAC method requires several hours of training with an ordinary PC and therefore incurs all the computational cost in the training phase, which is typically not critical for practical

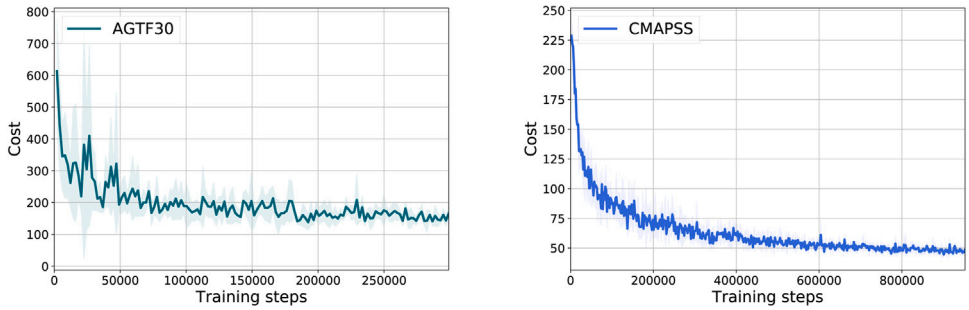


**Fig. 4.** The average calibration performance of on AGTF30 and CMAPSS task, where the shaded areas show the 1-SD confidence intervals over 5 random seeds. The $X$-axis indicates the total training steps, while $Y$-axis indicates the test return.

**Table 3**
Overview of the average time required for inference of a single sample with UKF, E2E, and CLAC approaches in seconds [s] for Case Study #1.

| Method | UKF | E2E | CLAC |
|---|---|---|---|
| Deployment time [s] | 6 | 4.2e−02 | **4.0e−02** |

**Table 4**
Overview of the inference performance (RMSE) under model bias (i.e. $F(w,\theta)\times\alpha$) and noise (i.e., $F(w,\theta)(1+N(0,\alpha_\eta))$ with UKF and CLAC approaches in Case Study #1.

| Model bias: $F(w,\theta)\times\alpha$ | | |
|---|---|---|
| Intensity | UKF | CLAC |
| $\alpha = 1.02$ | 2.04e−3 | **3.30e−04** |
| Model noise: $F(w,\theta)(1+N(0,\alpha_\eta))$ | | |
| Intensity | UKF | CLAC |
| $\alpha_\eta = 0.1$ | # | **4.22e−04** |

applications. For real-time applications, the main limiting factor is the deployment time. Therefore, in terms of computational cost, the proposed method has a clear advantage over UKF.

**Robustness to environment uncertainty.** Robustness to model inaccuracy is an important aspect in model calibration. It is also a well known limitation of model-based methods such as UKF. To evaluate the sensitivity of different approaches to inaccuracies in the models, we apply a model bias to the output of the dynamic system model (i.e., $F(w,\theta)\times\alpha$) to emulate an inadequacy of the system model structure (i.e., inaccurate simulator). We also consider a case where Gaussian noise is added to the dynamic system model, i.e., $F(w,\theta)+\eta$ where $\eta \sim N(0,\alpha_\eta)$. It is worth noticing that adding noise to the output of the simulator transforms the deterministic model into a stochastic system model.

From the RL perspective, the presence of an inaccurate simulator is known as *sim-to-real* transfer. In fact, *sim-to-real* is always a critical problem in reinforcement learning since the agent is trained in a simulated environment which may be different from the real world. In our case, we use a surrogate DNN model to accelerate the training. Therefore, we have an unavoidable error between the DNN surrogate model $\hat{x}_{t+1} = F(w_{t+1},\hat{x}_t,\theta_{t+1})$ and the engine physics-based model. Then, even in the case where noise is not added, the agent needs to make decisions with noisy DNN model outputs $\hat{x}_t$ at every time step $t$.

In order to test the trained policy under bias and noisy simulators, we tested two variants where we added a 2% fixed bias (i.e., $\alpha = 1.02$) and a Gaussian noise with a standard deviation of 10% of the model output (i.e., $F(w,\theta)(1+N(0,\alpha_\eta))$ with $\alpha_\eta = 0.1$) to the output of the DNN model. The results in Table 4 show that the policy obtained with the CLAC model provides a very good inference even under quite large uncertainty, demonstrating better robustness than the UKF, which failed to optimize a stable inference. The superior inference performance of the CLAC model under 2% fixed bias is visualized in Fig. 5.

**Scalability to more complex system and high dimensional model parameters** $\theta$. When the dimensionality of the physics-based model parameters $\theta$ increases, the complexity of inference increases as well. Due to the non-linear correlation between the degradation parameters and also between the degradation parameters and observations, the solution of the calibration problem in high dimensional spaces can lead to *confounding* solutions. In scenarios with noisy observations and systems with poor observability, the solution of inverse problems, such as UKF methods, might involve a spurious association of calibration factors that have similar system outputs. To test the scalability of our policies, we performed experiments on controlling 1, 2, and 4 degradation parameters in AGTF30 experiments (i.e. Case Study #2). Fig. 6 shows the inferred and ground truth traces of a four-dimensional $\theta$ in Case Study #2 with UKF (left) and CLAC (right) approaches. As in the previous plots, the $\theta$ values for 1315 fault intensities are stacked one after the other, thus generating a single time sequence. We can observe that the UKF solution does *confound* or *smear* the source of degradation. Moreover, as observed in Case Study #1, at the beginning of each fault combination, the predictions show large bias. Both of these issues are efficiently solved with the proposed CLAC method.

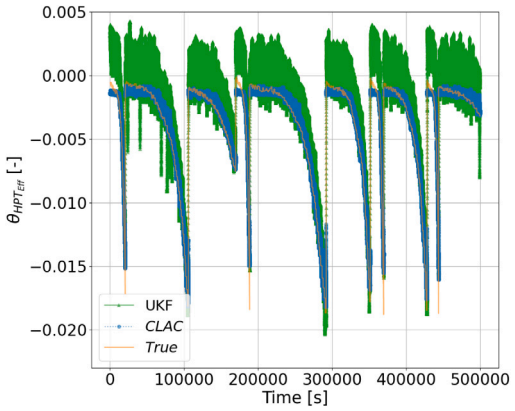


**Fig. 5.** Inferred and ground truth (orange squares) traces of $\theta$ in Case Study #1 with 2% model bias for UKF (green triangles) and CLAC (blue dots).

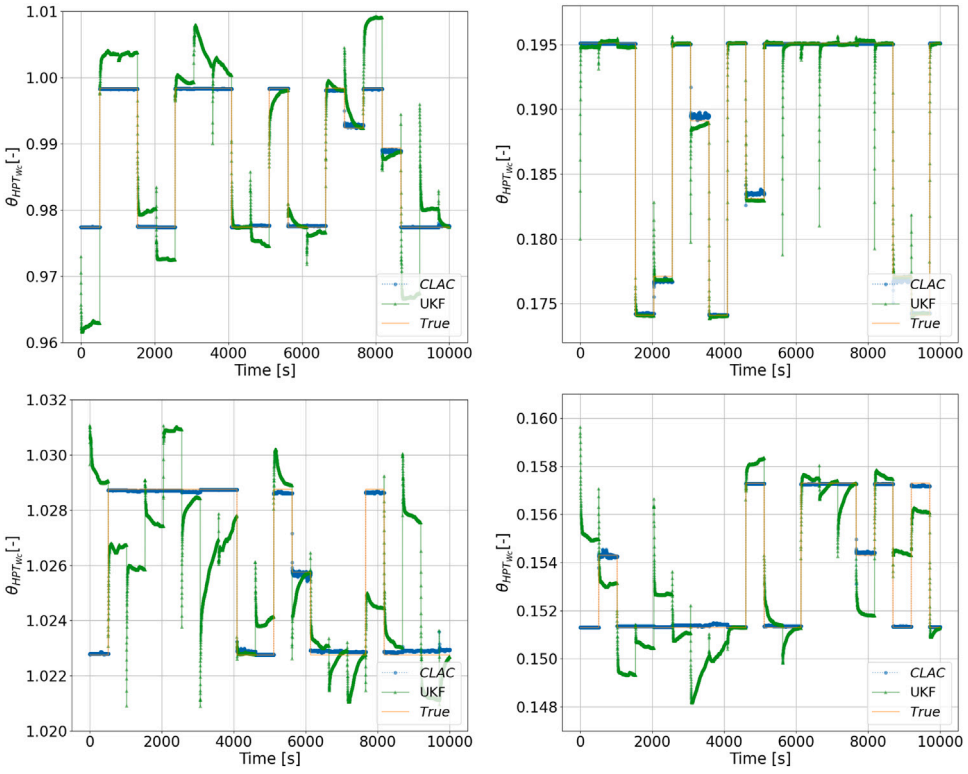


**Fig. 6.** Comparison to the ground truth in four $\theta$ parameters (HPT and LPT flow (i.e., Wc) and efficiency (i.e., Eff.)) and different degradation parameter settings. The orange solid line is the ground truth degradation parameter value in different trajectories. The blue dotted line is the policy's action with CLAC and the green solid line with triangles is the UKF prediction.

**Robustness to sensor noise.** In real scenarios, the observations are always noisy. Therefore, it is also important to obtain a policy that is robust to sensor noise. To evaluate this effect, we modelled the engine sensor noise and generated a noisy flight condition by adding Gaussian noise with an intensity of 70 db signal to noise ratio ($SNR_{db} = 70$) to the original flight condition. Table 5 shows the impact of noise on the inference performance of the UKF and CLAC methods. In this case, although our policy still shows good inference ability, UKF is more robust to sensor noise.

**Tracking accuracy.** We proposed to formulate the calibration problem as a tracking problem and use reinforcement learning to track the operational trajectories of the real systems (i.e., the observations) while being constrained to have a stable policy. Therefore, we evaluate the error between the observed real system response and the calibrated model output. Fig. 7 shows that our policies exhibit a good tracking ability for the model outputs. Table 6 provides a complete overview of the RMSE for each of the evaluated test cases. Although the CLAC framework demonstrates a good tracking ability in all the setups, the UKF achieves

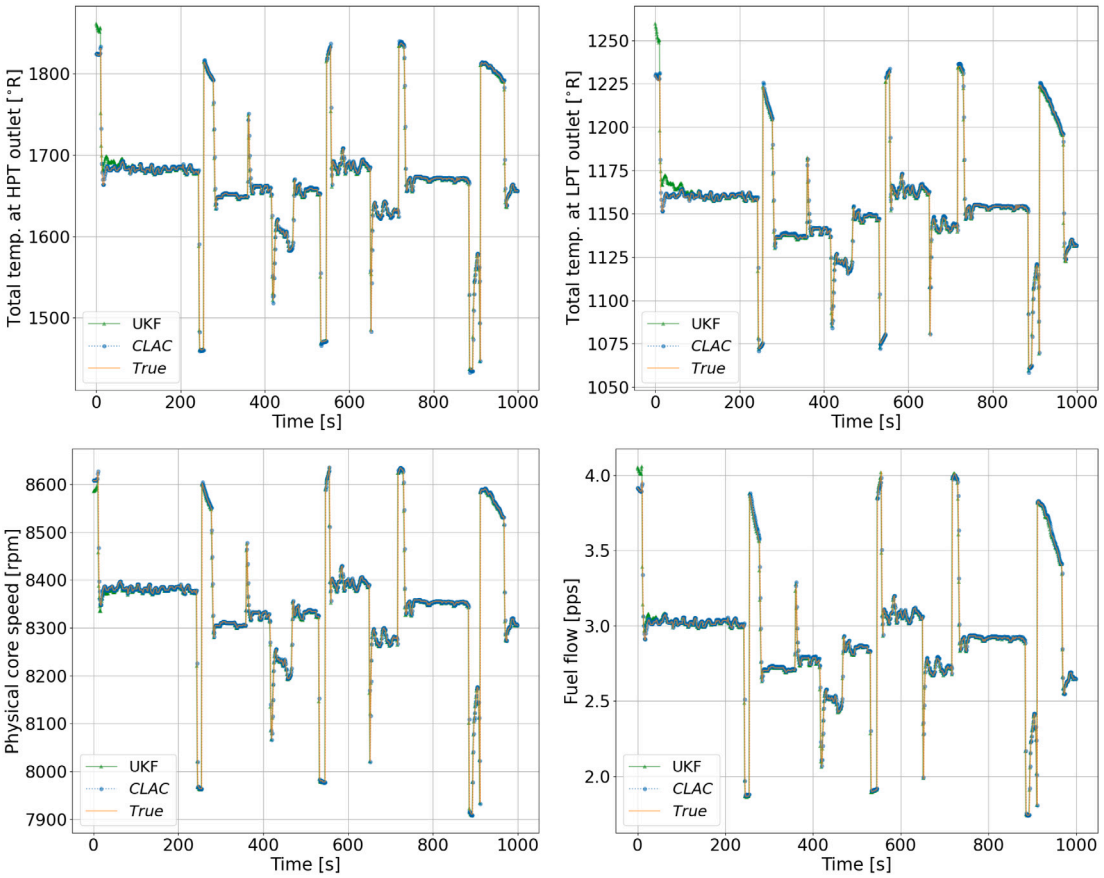


**Fig. 7.** Tracking with the proposed CLAC method (blue dots) and UKF (green triangles) on a subset of Case Study #1 for four sensor outputs. Ground truth is shown with the orange solid line. The CLAC policy is able to keep all these sensors on track.

**Table 5**
Overview of the inference performance under observation noise with UKF and CLAC approaches for complete test trajectories - System #1.

| Observation Noise: $x_s + \epsilon$ | | |
|---|---|---|
| Intensity | UKF | CLAC |
| $\text{SNR}_{db} = 70$ | **3.72e−04** | 7.18e−04 |

**Table 6**
Overview of the tracking performance given by RMSE with UKF and CLAC approaches in both case studies.

| Method | Case Study #1 | Case Study #2 |
|---|---|---|
| UKF | **0.62** | **1.78** |
| CLAC | 0.98 | 5.54 |

an even better tracking. This difference in the performance between UKF and CLAC is expected since in the RL agent is actually solving a more complicated problem. In particular, the current state contains the output of the DNN model $\hat{x}_t$ instead of the historical observation ($x_t$). As a result, small errors accumulate affecting the tracking performance. However it is worth point out that precisely this aspect ensures that the proposed policy action generalizes well to unseen degradation trajectories.

### *5.1. Ablation study*

**Comparison between LAC and CLAC algorithms** In this research, we propose to extend LAC to CLAC to improve the stability of the policy under noisy conditions. To demonstrate the benefit of the proposed extension, we compare the inference performance of both algorithms, LAC and CLAC, on Case Study #1. In the C-MAPSS experiments, the flight conditions are very diverse and the DNN model is not very accurate and is particularly noisy. Therefore, the DNN model may lead to an unstable policy. Fig. 8 shows

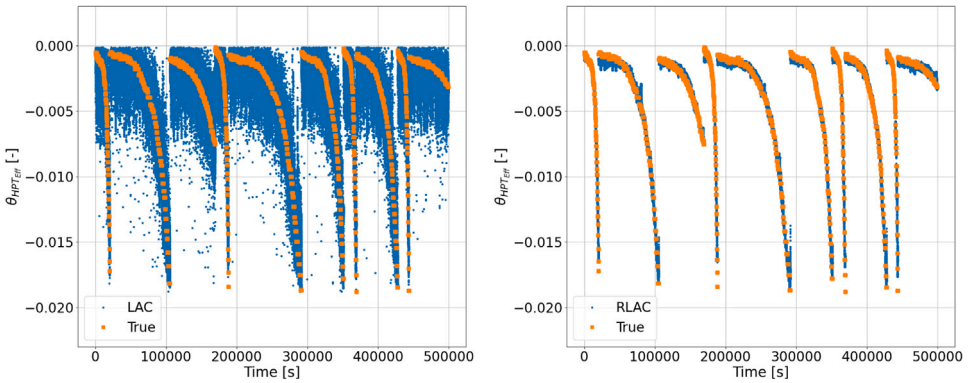


**Fig. 8. Left:** Inferred policy's actions with LAC algorithm (blue dots) and ground truth (orange squares) in Case Study #1. The trajectories of the ten units, stacked one after the other, are shown. SAC policy exhibits quite large variance. **Right:** Inferred policy's actions with CLAC algorithm (blue dots) and ground truth (orange squares). The CLAC policy shows a good stability and a very good match to the ground truth.

the policy's actions with the LAC and CLAC algorithm (orange squares) and ground truth (blue dots) for all the trajectories. CLAC demonstrates a significant reduction in the variance of the policy. Concretely, in terms of the RMSE metric, the LAC results in a RMSE of 1.3e−03 while the CLAC leads to an RMSE of 3.3e−04. Therefore, CLAC provides a 4× inference improvement.

## 6. Conclusions and future work

In this research, we proposed a maximum entropy reinforcement learning framework and the constrained Lyapunov-based actor–critic (CLAC) algorithm for model calibration. The proposed calibration methodology achieves an instantaneous, high inference accuracy and robustness that makes the proposed methodology applicable to noisy, and large-scale calibration problems in real-time. This capability was achieved purely on the basis of training in a simulation environment without any tedious sampling or computationally expensive solution of an inverse problem. Moreover, and in contrast to the end-to-end learning architectures, the proposed methodology only requires access to the model and the observations, eliminating the need for any ground truth calibration parameters for training. Overall, the proposed CLAC algorithm achieves more precise and faster inference than the prior state-of-the-art approaches while being more robust to system model uncertainty.

The proposed framework can be generally combined with various RL algorithms, or can even be extended to the meta RL [38,39] or hierarchical RL [40,41]. All our experiments are currently performed in a simulated environment. As a next step, we plan to evaluate the resulting policies on real industrial plants or robots.

Although the learning framework presented in the work is demonstrated in a model-based diagnostics task, it is applicable to any physics-based model, including those used in so-called "digital twins". Therefore, the results presented in this paper suggest a promising research direction in the field of model calibration. From the application perspective, the targeted model-based diagnostics problem was solved using exclusively a set of three deep neural networks. Therefore, the proposed framework is a paradigm shift in the field of model-based diagnostics. Starting with a model-based problem, we demonstrate that a clever arrangement of deep neural networks can learn both the relevant physics of a complex system and the inference techniques required for diagnostics. It is worth pointing out that the use of deep neural networks is very diverse (e.g., functioning as the surrogate of a physics-based system model or as an inference network in a decision-making problem). The proposed framework demonstrates the great potential of fusing physics-based and deep learning models.

## CRediT authorship contribution statement

**Yuan Tian:** Conceptualization, Methodology, Data curation, Software, Validation, Writing – original draft, Visualization, Investigation. **Manuel Arias Chao:** Conceptualization, Methodology, Data curation, Software, Validation, Writing – original draft, Visualization, Investigation. **Chetan Kulkarni:** Writing – review & editing. **Kai Goebel:** Writing – review & editing. **Olga Fink:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgement

The contributions of Yuan Tian, Manuel Arias Chao, and Olga Fink were funded by the Swiss National Science Foundation (SNSF) Grant no. PP00P2_176878. The contributions of Chetan Kulkarni were performed under NASA Ames Research Center, Contract No. 80ARC020D0010.

**Table B.7**
Weight coefficients and scaling parameters.

| Weight coefficients | |
|---|---|
| Weights | Expression |
| $W_0^m$ | $\frac{\lambda}{n_\theta+\lambda}$ |
| $W_0^c$ | $W_0^m + 1 - \alpha^2 + \beta$ |
| $W_i^c = W_i^m$ | $\frac{1}{2(n_\theta+\lambda)} \quad \forall i = 1, \ldots 2n_\theta$ |
| **Scaling parameters** | |
| Parameters | Values |
| $\gamma$ | $\sqrt{n_\theta + \lambda}$ |
| $\lambda$ | $\alpha^2(n_\theta + \kappa) - n_\theta$ |
| $\alpha$ | 1e−3 |
| $\beta$ | 2 |
| $\kappa$ | 0 |

## Appendix A. Training environment

### A.1. Case Study #1: A small fleet of Turbofan engines

Case Study #1 is the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dynamical model [42] that provides degradation trajectories of a small fleet comprising ten turbofan engines with unknown and different initial health conditions. The trajectories are given in the form of multivariate time-series of sensor readings (i.e., $[w, x]$). Real flight conditions ($w$), as recorded on board a commercial jet, were taken as input to the C-MAPSS model [43]. More details about the generation process can be found in [44].

### A.2. Case Study #2: A set of fault scenarios in Turbofan engines

Case Study #2 is the AGTF30 (Advanced Geared Turbofan 30k lbf) dynamical model [45] taking as input real flight conditions as recorded on board a commercial jet [43]. It can provide simulated condition monitoring trajectories of an advanced gas turbine during three flight profiles and multiple fault scenarios. Concretely, three different flight trajectories with a duration of 5000 s are considered. The system consists of concatenated time series of sensor readings (i.e., $[w, x_s] \in R^n$) resulting from faulty engine conditions. The fault conditions are induced and simultaneously affect four health-related model parameters representing model modifiers of the high pressure turbine (HPT) and low pressure turbine (LPT) flow and efficiency. A total of 1315 different fault scenarios are simulated by factorial design of a finite set of possible degradation intensities for each component. No additional noise was added to the model response since the flight conditions are already noisy.

## Appendix B. E2E and UKF settings

### B.1. E2E

To evaluate the different calibration methods under equivalent models, the E2E network is as also a MLP. In this way, we separate the effect of regularization in the form of model and learning strategies choice from other inductive bias in the form of choice of neural network type. The hidden layers have 100 units ($n^1 = n^2 = n^3 = 100$). The output layer has the dimension of the sensor reading vector (i.e. $n^L = n$). *ReLU* activation function was used throughout the hidden layers. For the output layer $\sigma^L = I$ is the identity. The resulting architecture is the result of a grid reach.

### B.2. UKF

The solution of the calibration problem with an UKF requires and state-update formulation where the state equation is modelled as a random walk. However, the estimation of the state (mean vector $\hat{\theta}^{(k)}$ and covariance matrix $P^{(k)}$) at each time steps is obtained with the UKF. Concretely, it resorts the standard UKF update process given by the algorithm 2. For clarity and completeness, the algorithm 2 and the corresponding UKF settings is added to the Appendix. A more detailed explanation of this problem formulation applied to the monitoring of gas turbine engines can be found in [17].

where the weight coefficients $W_0^m$, $W_0^c$ and $W_i^c$ and the corresponding scaling parameters are given in Table B.7.

The UKF algorithm required the definition of the diagonal covariance matrices $Q$ and $R$. We assumed the covariance matrices to be diagonal matrices with normalized standard deviation $r = 0.01$ and $q = 0.01$ (i.e., $Q = q^2\mathbf{I_n}$ and $R = r^2\mathbf{I_d}$ where $\mathbf{I_k}$ is the identity matrix of dimension $k$).

**Algorithm 2:** Unscented Kalman filter for health parameter estimation

**Estimation of health parameters**

**Input** : $\{w^{(k)}, x^{(k)}, \}_{k=1}^{m}$, $\theta_{ref}$, R and $P^{(0)} = Q$ & F

**Output**: $\hat{x}^{(k)}, \hat{\theta}^{(k)}, P^{(k)}$

**for** $k = 1 : m$ **do**

$\bar{P}^{(k)} = P^{(k-1)} + Q^{(k)}$

$\mathcal{S}^{(k-1)} = [\hat{\theta}^{(k-1)}, \hat{\theta}^{(k-1)} + \gamma\sqrt{\bar{P}^{(k)}}, \hat{\theta}^{(k-1)} - \gamma\sqrt{\bar{P}^{(k)}}]$ i.e. Sigma points (vectors)

$\mathcal{X}_i^{(k)} = F(w^{(k)}, \mathcal{S}_i^{(k-1)})$ for $i = 0, 1, \ldots, 2d$ Propagate $\mathcal{S}$ through the non-linear function $F$

$\hat{x}_s^{(k)} = \sum_{i=0}^{2d} W_i^m \mathcal{X}_i^{(k)}$

$r^{(k)} = x_s^{(k)} - \hat{x}_s^{(k)}$

$P_y^{(k)} = \sum_{i=0}^{2d} W_i^c (\mathcal{X}_i^{(k)} - \hat{x}_s^{(k)})(\mathcal{X}_i^{(k)} - \hat{x}_s^{(k)})^T + R$

$P_{\theta y}^{(k)} = \sum_{i=0}^{2d} W_i^c (\mathcal{S}^{(k-1)} - \hat{\theta}^{(k-1)})(\mathcal{X}_i^{(k)} - \hat{x}_s^{(k)})^T$

$K^{(k)} = P_{\theta y}^{(k)} {P_y^{(k)}}^{-1}$

$\hat{\theta}^{(k)} = \hat{\theta}^{(k)} - K^{(k)} r^{(k)}$

$P^{(k)} = \bar{P}^{(k)} - K^{(k)} P_y^{(k)} {K^{(k)}}^T$

**end**

**end**

## References


[1] M.C. Kennedy, A. O'Hagan, Bayesian calibration of computer models, J. R. Stat. Soc. Ser. B Stat. Methodol. 63 (3) (2001) 425–464, http://dx.doi.org/10.1111/1467-9868.00294, URL http://doi.wiley.com/10.1111/1467-9868.00294.

[2] A.H. Elsheikh, V. Demyanov, R. Tavakoli, M.A. Christie, M.F. Wheeler, Calibration of channelized subsurface flow models using nested sampling and soft probabilities, Adv. Water Resour. 75 (2015) 14–30, http://dx.doi.org/10.1016/j.advwatres.2014.10.006.

[3] B. Sansó, C.E. Forest, D. Zantedeschi, Inferring Climate System Properties Using a Computer Model, Tech. rep. 1, 2008, pp. 1–38, URL http://www.ams.ucsc.edu/{~}danielz/.

[4] D.A. Henderson, R.J. Boys, K.J. Krishnan, C. Lawless, D.J. Wilkinson, Bayesian emulation and calibration of a stochastic computer model of mitochondrial DNA deletions in substantia nigra neurons, J. Amer. Statist. Assoc. 104 (485) (2009) 76–87, http://dx.doi.org/10.1198/jasa.2009.0005.

[5] C. Rutter, D. Miglioretti, J. Savarino, Bayesian calibration of microsimulation models, J. Amer. Statist. Assoc. 104 (488) (2009) 1338–1350, http://dx.doi.org/10.1198/jasa.2009.ap07466.

[6] S. Liu, A. Borovykh, L.A. Grzelak, C.W. Oosterlee, A neural network-based framework for financial model calibration, J. Math. Ind. 9 (1) (2019) 1–28, http://dx.doi.org/10.1186/s13362-019-0066-7, arXiv:1904.10523.

[7] Z.C. Deng, J.N. Yu, L. Yang, An inverse problem of determining the implied volatility in option pricing, J. Math. Anal. Appl. 340 (1) (2008) 16–31, http://dx.doi.org/10.1016/j.jmaa.2007.07.075.

[8] A. Kangasrääsiö, J.P.P. Jokinen, A. Oulasvirta, A. Howes, S. Kaski, Parameter inference for computational cognitive models with approximate Bayesian computation, Cogn. Sci. 43 (6) (2019) http://dx.doi.org/10.1111/cogs.12738, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cogs.12738.

[9] N.C. Kumar, A.K. Subramaniyan, L. Wang, G. Wiggs, Calibrating transient models with multiple responses using bayesian inverse techniques, in: Proceedings of the ASME Turbo Expo, Vol. 7 A, American Society of Mechanical Engineers Digital Collection, 2013, http://dx.doi.org/10.1115/GT2013-95857.

[10] D. Higdon, J. Gattiker, B. Williams, M. Rightley, Computer model calibration using high-dimensional output, J. Amer. Statist. Assoc. 103 (482) (2008) 570–583, URL http://www.jstor.org/stable/27640080.

[11] I. Roychoudhury, V. Hafiychuk, K. Goebel, Model-based diagnosis and prognosis of a water recycling system, in: 2013 IEEE Aerospace Conference, 2013, pp. 1–9.

[12] J.L. Crassidis, J.L. Junkins, Optimal Estimation of Dynamic Systems, Second Edition (Chapman & Hall/CRC Applied Mathematics & Nonlinear Science), second ed., Chapman & Hall/CRC, 2011.

[13] J. Sacks, W.J. Welch, J.S.B. Mitchell, P.W. Henry, Design and experiments of computer experiments, in: Statistical Science, Vol. 4, 1989, pp. 409–423, http://dx.doi.org/10.2307/2245858.

[14] M. Arias Chao, D.S. Lilley, P. Mathé, V. Schloßhauer, Calibration and uncertainty quantification of gas turbine performance models, in: Proceedings of the ASME Turbo Expo, Vol. 7A, 2015, http://dx.doi.org/10.1115/gt2015-42392, V07AT29A001.

[15] S.J. Julier, J.K. Uhlmann, New extension of the Kalman filter to nonlinear systems, in: Signal Processing, Sensor Fusion, and Target Recognition VI, Vol. 3068, 1997, p. 182, http://dx.doi.org/10.1117/12.280797.

[16] R. Turner, C.E. Rasmussen, Model based learning of sigma points in unscented Kalman filtering, in: Proceedings of the 2010 IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2010, 2010, pp. 178–183, http://dx.doi.org/10.1109/MLSP.2010.5589003.

[17] S. Borguet, Variations on the Kalman Filter for Enhanced Performance Monitoring of Gas Turbine Engines (Phd thesis), Université de Liège, 2012.

[18] N. Kantas, A. Doucet, S.S. Singh, J. Maciejowski, N. Chopin, On particle methods for parameter estimation in state-space models, Statist. Sci. 30 (3) (2015) 328–351, http://dx.doi.org/10.1214/14-STS511.

[19] Y. Zhang, L. Guo, B. Gao, T. Qu, H. Chen, Deterministic promotion reinforcement learning applied to longitudinal velocity control for automated vehicles, IEEE Trans. Veh. Technol. 69 (1) (2020) 338–348.

[20] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, I. Palunko, Reinforcement learning for control: Performance, stability, and deep approximators, Annu. Rev. Control (2018).

[21] V. Kumar, A. Gupta, E. Todorov, S. Levine, Learning dexterous manipulation policies from experience and imitation, 2016, arXiv preprint arXiv:1611.05095.

[22] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, M. van de Panne, Iterative reinforcement learning based design of dynamic locomotion skills for cassie, 2019, arXiv preprint arXiv:1903.09537.

[23] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, M. Hutter, Learning agile and dynamic motor skills for legged robots, Science Robotics 4 (26) (2019) eaau5872.

[24] R.S. Sutton, A.G. Barto, R.J. Williams, Reinforcement learning is direct adaptive optimal control, IEEE Control Syst. Mag. 12 (2) (1992) 19–22.
[25] M. Han, Y. Tian, L. Zhang, J. Wang, W. Pan, H∞ model-free reinforcement learning with robust stability guarantee, 2019, arXiv preprint arXiv:1911.02875.
[26] R.S. Sutton, A.G. Barto, et al., Introduction to Reinforcement Learning, Vol. 135, MIT press Cambridge, 1998.
[27] B.D. Ziebart, Modeling purposeful adaptive behavior with the principle of maximum causal entropy, 2010.
[28] L. Ljung, S. Gunnarsson, Adaptation and tracking in system identification-A survey, Automatica 26 (1) (1990) 7–21, http://dx.doi.org/10.1016/0005-1098(90)90154-A.
[29] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al., Soft actor-critic algorithms and applications, 2018, arXiv preprint arXiv:1812.05905.
[30] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[31] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.
[32] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al., Soft actor-critic algorithms and applications, 2018, arXiv preprint arXiv:1812.05905.
[33] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proc. Icml, Vol. 30, 2013, p. 1.
[34] D.P. Kingma, J.L. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015, arXiv:1412.6980.
[35] X. Glorot, Y. Bengio, Understanding the Difficulty of Training Deep Feedforward Neural Networks, Tech. rep., 2010, URL http://www.iro.umontreal.
[36] M. Han, L. Zhang, J. Wang, W. Pan, Actor-critic reinforcement learning for control with stability guarantee, IEEE Robot. Autom. Lett. 5 (4) (2020) 6217–6224.
[37] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep reinforcement learning that matters, 2019, arXiv:1709.06560.
[38] K. Rakelly, A. Zhou, D. Quillen, C. Finn, S. Levine, Efficient off-policy meta-reinforcement learning via probabilistic context variables, 2019, arXiv preprint arXiv:1903.08254.
[39] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1126–1135.
[40] T.G. Dietterich, Hierarchical reinforcement learning with the MAXQ value function decomposition, J. Artificial Intelligence Res. 13 (2000) 227–303.
[41] A.G. Barto, S. Mahadevan, Recent advances in hierarchical reinforcement learning, Discrete Event Dyn. Syst. 13 (1–2) (2003) 41–77.
[42] D.K. Frederick, J.A. Decastro, J.S. Litt, User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS), Tech. rep., 2007.
[43] NASA, DASHlink - Flight Data For Tail 687, 2012, URL https://c3.nasa.gov/dashlink/.
[44] M. Arias Chao, C.S. Kulkarni, K. Goebel, O. Fink, Aircraft engine run-to-failure dataset under real flight conditions for prognostics and diagnostics, 2021, Data 2021, 6(1), 5, URL https://doi.org/10.3390/data6010005.
[45] J.W. Chapman, J.S. Litt, Control design for an advanced geared turbofan engine, in: 53rd AIAA/SAE/ASEE Joint Propulsion Conference, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2017, http://dx.doi.org/10.2514/6.2017-4820, URL https://arc.aiaa.org/doi/10.2514/6.2017-4820.